Extractive Text Summarization with Deep Learning

Senior Project Final Report

By: Garrett Chan
Advisor: Dr. Aaron Keen
Repository: https://github.com/ggchan0/TLDR

Table of Contents:

1. Introduction:

Problem Description:

“Textual information in the form of digital documents quickly accumulates to huge amounts of data. Most of this large volume of documents is unstructured: it is unrestricted and has not been organized into traditional databases. Processing documents is therefore a perfunctory task, mostly due to the lack of standards” [1]. Consequently, there exists a need for a mechanism to reduce the size of the text, create some sort of structure to it, and process the document to make it readable for the user. This project aims to address all three concerns to the problem with the growing amount of textual information in the world with extractive text summarization.

Solution:

Currently, there exist automatic text summarizers that all use differing techniques. One popular approach is to derive statistical information from word frequency and distribution and to select sentences that score the highest. Another approach utilizes this information alongside sentence placement, cue words, and title and heading words present in the sentence. Techniques via deep learning and machine learning have made breakthroughs in abstractive summarization of the text. The approach in this project utilizes deep learning to determine whether a not a sentence, based off of several key features from the text, should be apart of the summarization. By tokenizing the text into paragraphs and sentences, and analyzing each sentence through a neural network, one might be able to create a comprehensive summary.

2. Background:

Key Terms:

Text Summarization:

Automatic text summarization “is the process of distilling the most important information from a source (or sources) to produce an abridged version for a particular user (or users) and task (or tasks)” [2]. The two types of summarization are abstractive and extractive text summarization. Abstractive text summarization aims to generate a summary that paraphrases the original text and is easily readable by a human. Extractive text summarization aims to pull words, phrases, or sentences from the original text to create a summary. The approach provided in this project utilizes extractive summarization.

Deep Learning:

Deep learning utilizes neural networks to simulate how the human brain works. These neural networks range from small networks of a few neurons to very large ones with thousands of neurons in order to process large amounts of data. The goal is to “make learning algorithms much better and easier to use” and to “make revolutionary advances in machine learning and AI” [3].

Neural Network:

Neural networks are computing systems that are inspired by the structure of the human brain and nervous system. Neural networks take a vector of features as its input and can output a single result or a vector of data.

Feature:

Features are properties or characteristics of some sort of measurement or observation. In the context of this project, features are differing types of data extracted from the text. Features are typically combined into a vector of data to be fed into a neural network.

Natural Language Processing:

Natural language processing (NLP) is concerned with how computers process the natural language of humans. Libraries built for NLP are utilized for feature extract in this project.

Tokenization:

Tokenization is the process by which a system breaks up a particular bit of data into pieces, called tokens. In the context of the project, the input text will be tokenized into paragraphs, and paragraphs will be tokenized into individual sentences. Individual sentences are tokenized into words for more processing.

Keyword Extraction:

Keyword extraction involves listing the key phrases or terms that are involved in presenting the most relevant information contained in the document. Keywords will be used in feature extraction for this project.

Technologies Used:

Keras:

Keras is a high-level neural networks API that utilizes Tensorflow as a backend and is used for the deep learning of this project [4].

NLTK (Natural Language Toolkit):

The Natural Language Toolkit collection of open source Python modules involved with Natural Language Processing [5]. It is used for tokenizing and extracting important features from the text. It also does part-of-speech tagging, which is useful for identifying proper nouns and statistics, two features which are used in summarizing the text.

TextBlob:

TextBlob is a library for processing text data, and provides simple set of functions for Natural Language Processing [6]. TextBlob is used for sentiment analysis in the project. The output returns a value from -1.0 to 1.0. The closer to -1.0, the more negative the sentiment is. The converse is true for higher sentiment scores as values closer to 1.0 signify positive sentiment.

Rake (Rapid Automatic Keyword Extraction):

The RAKE keyword extraction algorithm determines keywords or phrases in text by analyzing the frequency of words and their occurrence along other words. This library outputs keywords/phrases alongside their weighted score. The higher the weight, the more important or relevant the keyword/phrase is.

Python3:

Python is an interpreted programming language. It is the primary language used in the project as Keras and all of the NLP libraries are built to be used by Python. The specific version that was used is Python version 3.4.3.

Flask:

Flask is a framework for Python to easily create web-based APIs. Flask is used to construct the API and allow incoming connections to interface with the model.

IBM Watson Natural Language Understanding:

IBM Watson’s Natural Language Understanding provides a suite of utilities implementing NLP algorithms to extract features from text. This tool is used in the project primarily to evaluate the sentiment performance of summarized text. The output consists of the emotions of fear, anger, joy, sadness, and disgust.

Text Features:

Six features were pulled from each sentence to be fed into the neural network. Based upon these features, the neural network made a decision about whether to include the sentence or not.

Sentiment Difference:

The sentiment difference between the sentence in question and the original text is a feature for the neural network. The motivation is that sentences with sentiment that are similar to the overall text belong in the text. This helps to filter out outlier sentences that convey lots of polarizing emotion, but not information. This is a positive number of the absolute value of the difference between the sentence and the whole text. The equation is as follows:

$$|\textit{Sentiment score of entire text} - \textit{sentiment score of the sentence}|$$

Proper Noun Ratio:

The number of proper nouns in each sentence was calculated and divided by the total number of words in the sentence to give the proper noun ratio. Sentences with lots of proper nouns in them are more likely to be pivotal to the summarization than sentences with less or no proper nouns in them. This value varies from 0 to 1.0. The equation is as follows:

$$|\textit{\# of proper nouns in sentence / length of sentence}|$$

Stat Ratio:

Similar to the proper noun ratio, the statistic ratio gave a proportion for how many statistics, numbers, and data were part of a sentence. Sentences with statistics in them give important information that should typically be included as part of the summarization. This value varies from 0 to 1.0.

$$|\#\ \textit{of statistics in sentence} / \textit{length of sentence}|$$

Keyword Score:

The keyword score consists of the summation of the weights for all keywords/phrases in a sentence. Not every word will have a particular weight to it, but words with weights are summed up per sentence. Sentences with more keywords indicate that the sentence is important. Consequently, the higher the keyword score, the more likely the sentence will be used in the summary.

Sentence Length:

The sentence length is used as one of the features for analyzing the text. The sentence length score is merely the number of words per each sentence. Though simple, one might conjecture that the more words that are in a sentence, the more important the sentence is.

Sentence Position:

Sentences that are at the beginning or the end of paragraphs are considered more important. The sentence position value is calculated as follows:

$$|\ (\textit{cosine}(\textit{sentence number}) / \textit{number of sentences in the current paragraph}) * \textit{pi}\ |$$

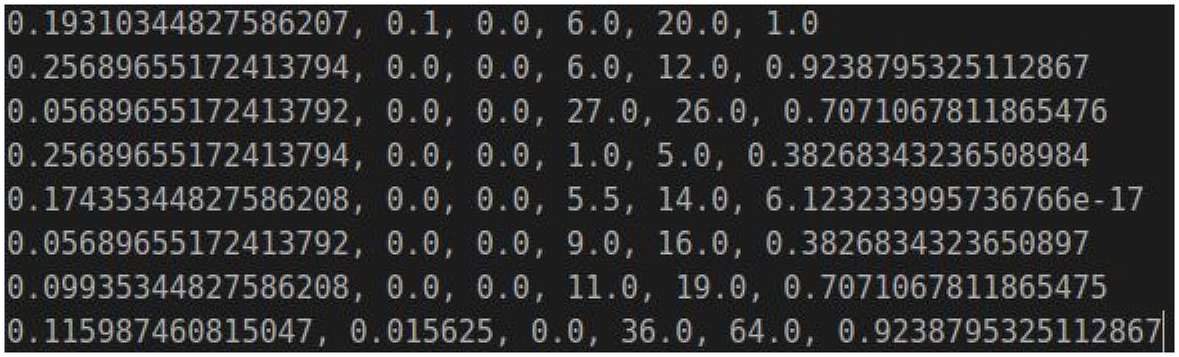
3. Implementation:

Feature Extraction:

To analyze the features from the text to be fed into the neural network, a feature extractor was created. The feature extractor tokenizes the text into paragraphs. It iterates through each paragraph, further tokenizing each one into a collection of sentences for feature analysis.

The NLTK is used to tokenize each sentence into words so that the length could be calculated. The library also gives numbers for the amount of proper nouns and statistics in each sentence. TextBlob is used to get the sentiment for the entire text, as well as each sentence. Next, the RAKE module was integrated to give a listing of the keyword phrases and scores in the text. Sentences were tokenized into words, which were then checked to see if they were part of the list of keywords. If a word is a keyword, the sentences keyword score is increased by the weight of the keyword. Finally, as the feature extractor iterates through each sentence in the paragraph, it assigns a sentence position score based on the number of the sentence within the paragraph, and the total number of sentences that the paragraph is comprised of.

Concatenating all six numbers gives a vector of inputs to be fed into the neural network. The ordering of the numbers is arbitrary as the neural network can take the vector in any order. However, the format must be consistent among all vectors.



```
0.19310344827586207, 0.1, 0.0, 6.0, 20.0, 1.0
0.25689655172413794, 0.0, 0.0, 6.0, 12.0, 0.9238795325112867
0.05689655172413792, 0.0, 0.0, 27.0, 26.0, 0.7071067811865476
0.25689655172413794, 0.0, 0.0, 1.0, 5.0, 0.38268343236508984
0.17435344827586208, 0.0, 0.0, 5.5, 14.0, 6.123233995736766e-17
0.05689655172413792, 0.0, 0.0, 9.0, 16.0, 0.3826834323650897
0.09935344827586208, 0.0, 0.0, 11.0, 19.0, 0.7071067811865475
0.115987460815047, 0.015625, 0.0, 36.0, 64.0, 0.9238795325112867
```

Vectors of the features extracted from the text in the order of sentiment difference, proper noun ratio, statistic ratio, keyword score, sentence length, and sentence position.

Training the Model:

The model was trained using samples of data from Reddit ELI5 (explain like I'm five) posts. These posts explain a particular topic or answer a question using simple examples and vocabulary. The text was directly taken off of reddit via the Reddit API. Sentences that I thought were crucial to understanding the post were marked. The features were then extracted and the neural network was trained under supervision.

While text summarization is usually applied to large volumes of text and not short reddit posts, the model can still be trained sufficiently. Since feature extraction happens per paragraph, longer texts don't necessarily indicate better data for the neural network. Due to time constraints on generating data, only 100 training samples were used.

The neural network consists of six input neurons, twelve hidden neurons, and an output layer of two neurons. Several different neural network configurations were tested, from having only six neurons in the hidden layer, to having an additional hidden layer. The final configuration yielded the most accurate fit as measured by Keras. The activation functions for the input and hidden layer were RELU, and the output layer used softmax. Since the dataset was very small, training the model was quick. It underwent 1000 epochs of training.

Interfacing with the Model:

The model was trained and saved into a model file that can be loaded on demand. For testing purposes, a python script to run summarization on an input file was created. In production however, a Flask API was created to accept HTTP requests to summarize text. It supports GET requests with the text to be summarized as the payload and returns JSON containing the summary.

One additional functionality that was added was N-summarization. The idea is to feed the text into the neural network more than once, or N times. By feeding the output of each run back into the neural network, the text can be further reduced.

Creating the Summary:

The output of the neural network is a list of confidence value from 0 to 1. Each number corresponds to a sentence in the text. Sentences that the neural network suggests should be in the summary return higher values. For the purposes of summarization, sentences that received a score of greater than 0.8 were included in the summarization.

4. Evaluating Performance:

Performance Indicators:

Performance indicators for the summarizer consists of several calculated values. The first is the reduction score, which measures how much the original text was summarized. This is the most important metric, as the goal of the summarizer is to reduce the text as much as possible, while keeping key facets of information. The next feature is the percent of weighted keywords included. If there are more relevant keywords in the text, the better the summary should be. Finally, IBM Watson's Natural Language Understanding API was used to give a more comprehensive sentiment breakdown. The sentiment difference of all emotions between the original and summarized text was calculated and displayed.

Control Dataset:

To provide good comparisons, several control summarizations were created.

First Sentence Summarization

First sentence summarization is inspired by the idea that the first sentence of the paragraph is the most important one. This control summarization takes the first sentence from every paragraph and concatenates them into a summary.

Last Sentence Summarization

Last sentence summarization does the opposite of first sentence summarization by taking the last sentence of every paragraph and forming a summary out of them.

First and Last Sentence Summarization

This method combines the first sentence and last sentence summarization, with the hope that the most important sentences from each paragraph are taken.

Every Other Sentence Summarization

Every other sentence summarization takes every other sentence from the text and creates a summary out of it. This method arbitrarily takes sentences to create a ~50% reduction in text.

Every Other Sentence in Paragraph Summarization

This method does the same as every other sentence summarization, but on a per paragraph basis.

Results:

Despite the lack of training data, the model proved to adequately reduce text, especially if used in N-summarization mode.

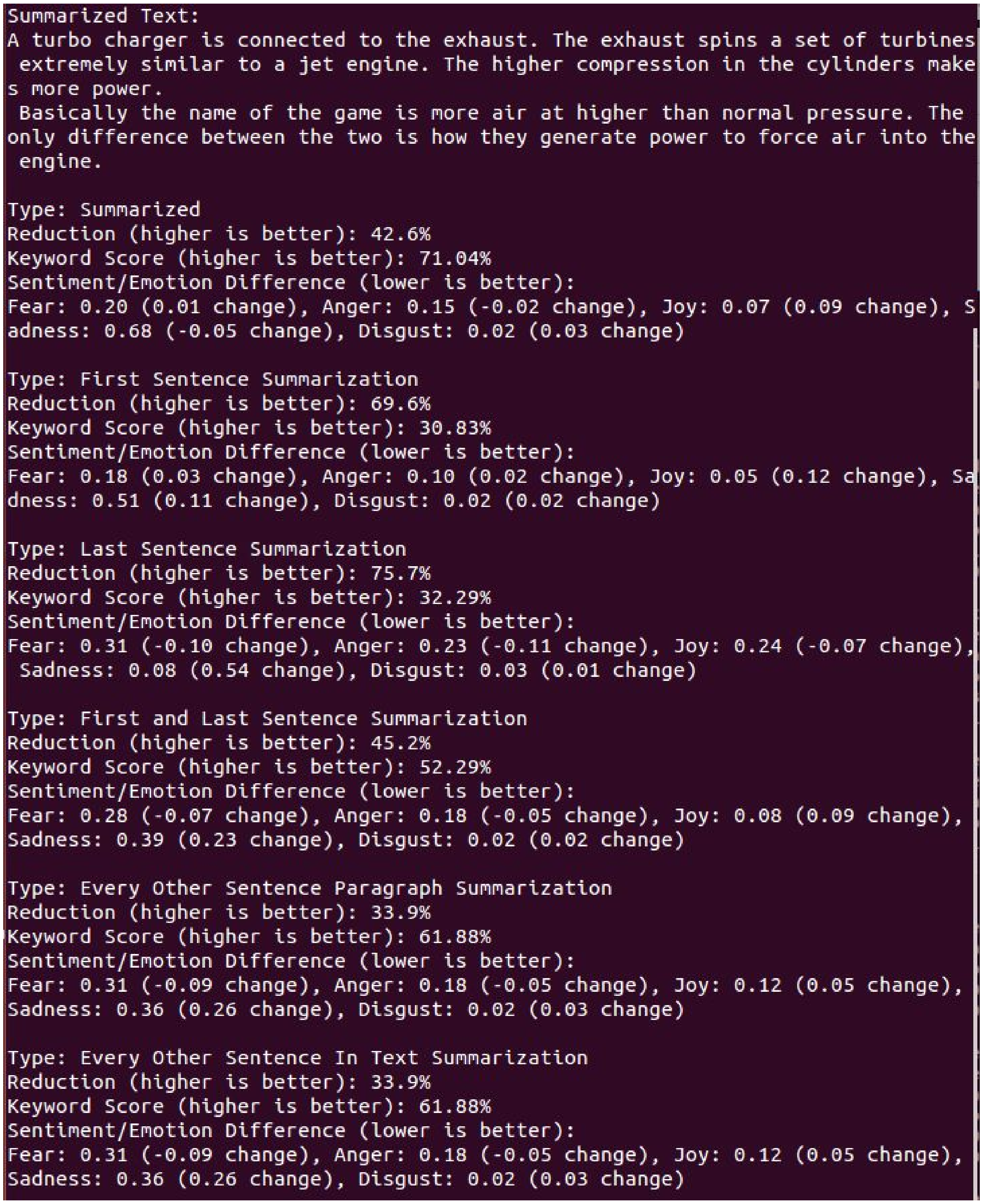
	Summary N = 1	Summary N = 3	First	Last	First and Last	Every Other Sentence	Every Other Sentence in Paragraph
Long ELI5							
Reduction Score	25.60%	44.20%	86.50%	84.90%	73%	48.20%	48.10%
Keyword Score	82.75%	65.60%	19.98%	19.77%	34.58%	57.25%	59.62%
Short ELI5							
Reduction Score	42.60%	58.30%	69.60%	75.70%	45.20%	33.90%	33.90%
Keyword Score	71.04%	57.50%	30.83%	32.29%	52.29%	61.88%	61.88%
Very Short ELI5							
Reduction Score	31.90%	31.90%	92.90%	85.10%	78%	48.90%	48.90%
Keyword Score	67.86%	67.86%	3.90%	26.62%	30.52%	50%	50%

Average							
	33.37%	44.80%	83.00%	81.90%	65.40%	43.67%	43.63%
	73.88%	63.65%	18.24%	26.23%	39.13%	56.38%	57.17%

Example data across three different textfiles

Interestly enough, in N-summarization mode, the less text there is to summarize, the less effective at reducing text it becomes. For a reddit post with one paragraph, the N=3 summarization gave the same result as just one pass through the neural network. For a larger text that spanned through two reddit posts, the effectiveness of N summarization dropped after N=4. There appears to be a correlation between the size of the text, and the highest value of N. The tradeoff is that having a large N value for a large text takes a lot of computation.

Sample Output:



```
Summarized Text:
A turbo charger is connected to the exhaust. The exhaust spins a set of turbines
 extremely similar to a jet engine. The higher compression in the cylinders make
s more power.
 Basically the name of the game is more air at higher than normal pressure. The
only difference between the two is how they generate power to force air into the
 engine.

Type: Summarized
Reduction (higher is better): 42.6%
Keyword Score (higher is better): 71.04%
Sentiment/Emotion Difference (lower is better):
Fear: 0.20 (0.01 change), Anger: 0.15 (-0.02 change), Joy: 0.07 (0.09 change), S
adness: 0.68 (-0.05 change), Disgust: 0.02 (0.03 change)

Type: First Sentence Summarization
Reduction (higher is better): 69.6%
Keyword Score (higher is better): 30.83%
Sentiment/Emotion Difference (lower is better):
Fear: 0.18 (0.03 change), Anger: 0.10 (0.02 change), Joy: 0.05 (0.12 change), Sa
dness: 0.51 (0.11 change), Disgust: 0.02 (0.02 change)

Type: Last Sentence Summarization
Reduction (higher is better): 75.7%
Keyword Score (higher is better): 32.29%
Sentiment/Emotion Difference (lower is better):
Fear: 0.31 (-0.10 change), Anger: 0.23 (-0.11 change), Joy: 0.24 (-0.07 change),
 Sadness: 0.08 (0.54 change), Disgust: 0.03 (0.01 change)

Type: First and Last Sentence Summarization
Reduction (higher is better): 45.2%
Keyword Score (higher is better): 52.29%
Sentiment/Emotion Difference (lower is better):
Fear: 0.28 (-0.07 change), Anger: 0.18 (-0.05 change), Joy: 0.08 (0.09 change),
Sadness: 0.39 (0.23 change), Disgust: 0.02 (0.02 change)

Type: Every Other Sentence Paragraph Summarization
Reduction (higher is better): 33.9%
Keyword Score (higher is better): 61.88%
Sentiment/Emotion Difference (lower is better):
Fear: 0.31 (-0.09 change), Anger: 0.18 (-0.05 change), Joy: 0.12 (0.05 change),
Sadness: 0.36 (0.26 change), Disgust: 0.02 (0.03 change)

Type: Every Other Sentence In Text Summarization
Reduction (higher is better): 33.9%
Keyword Score (higher is better): 61.88%
Sentiment/Emotion Difference (lower is better):
Fear: 0.31 (-0.09 change), Anger: 0.18 (-0.05 change), Joy: 0.12 (0.05 change),
Sadness: 0.36 (0.26 change), Disgust: 0.02 (0.03 change)
```

A sample output from the text summarizer with metrics

Sample Summarization:

Below is an example summarization from the original text.

Original:

A turbo charger is connected to the exhaust. The exhaust spins a set of turbines extremely similar to a jet engine. The air gets compressed in the turbines and is then forced out with greater force and is then rammed into the engine. The higher compression in the cylinders makes more power. A supercharger is connected to the engine itself and uses that power to force more air into the engine at higher pressure making more power. Basically the name of the game is more air at higher than normal pressure. The only difference between the two is how they generate power to force air into the engine.

Summarized (N = 1):

A turbo charger is connected to the exhaust. The exhaust spins a set of turbines extremely similar to a jet engine. The higher compression in the cylinders makes more power.

Basically the name of the game is more air at higher than normal pressure. The only difference between the two is how they generate power to force air into the engine.

5. Concluding Thoughts:

Overall, the project was an incredible learning experience. Prior to taking on the project, I had no experience in machine learning, deep learning, or neural networks. Since deep learning is a field growing at an incredible rate, I'm glad that I took on this project.

Several improvements can be made to the project. First, I would make to the project would be generating more training data. A sample of 100 was able to provide good results, but having a larger amount of training data would definitely make the summarizer more accurate. I would also add in more features to be fed into the neural network as a vector of six features isn't as comprehensive as it can be. To make the experiment better, I would not only compare my results to the control summarizations, but to text summarizers on the web and see how the deep learning approach compares.

6. References:

1. Juan-Manuel Torres-Moreno. 2014. Automatic Text Summarization. 1 (Ed.). Wiley-ISTE, London, UK

2. Inderjeet Mani. 1999. Advances in Automatic Text Summarization. Mark T. Maybury (Ed.). MIT Press, Cambridge, MA, USA.

3. Andrew Ng. 2013. Self-Taught Learning and Unsupervised Feature Learning. Video (13 May 2013). Retrieved June 5, 2018 from https://www.youtube.com/watch?v=n1ViNeWhC24

4. Keras. Keras: The Python Deep Learning Library. Retrieved from https://keras.io/.

5. NLTK. Natural Language Toolkit. Retrieved from https://www.nltk.org/.

6. TextBlob. TextBlob: Simplified Text Processing. Retrieved from http://textblob.readthedocs.io/en/dev/.